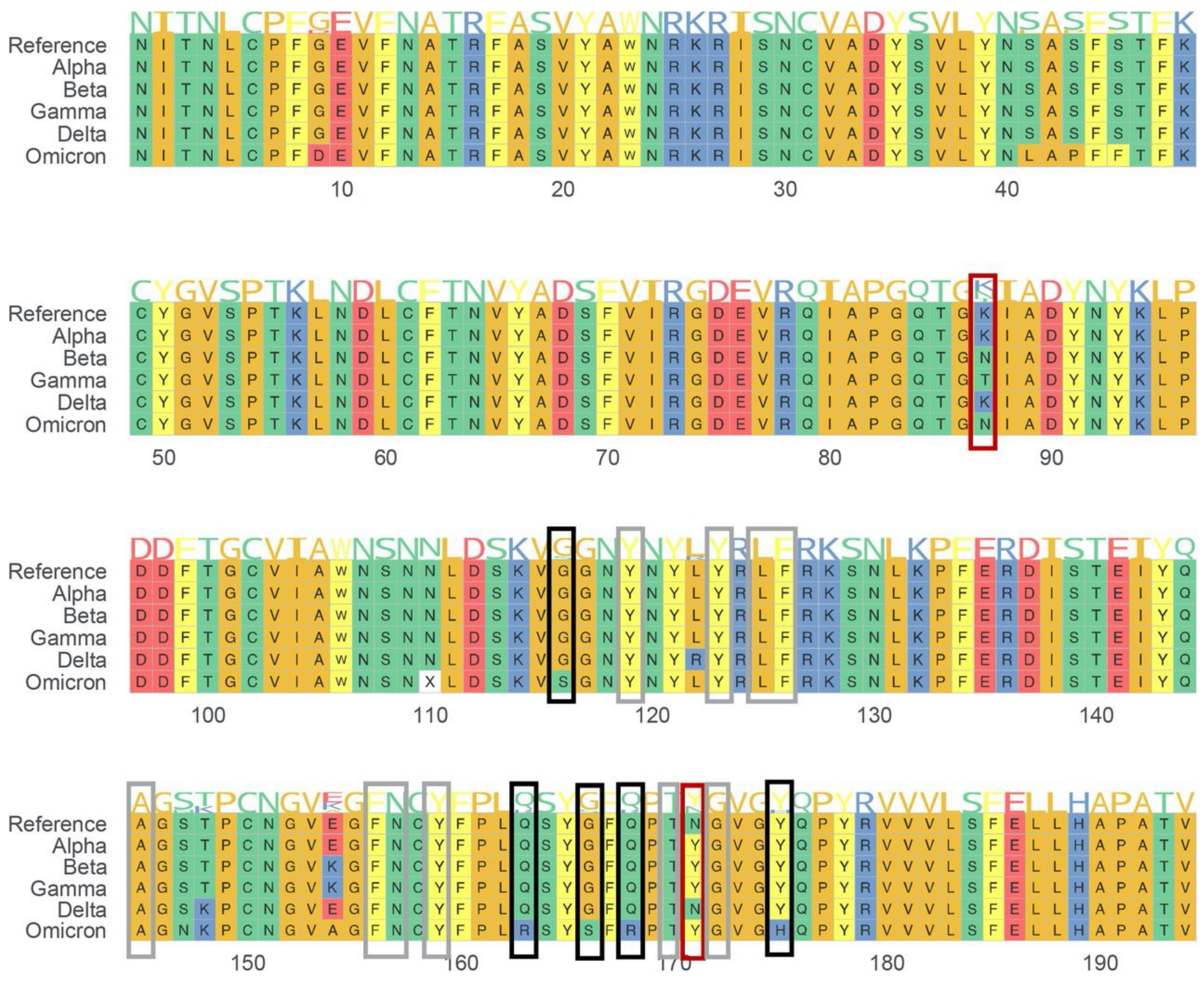
Multiple Sequence Alignment

Multiple Alignment Example



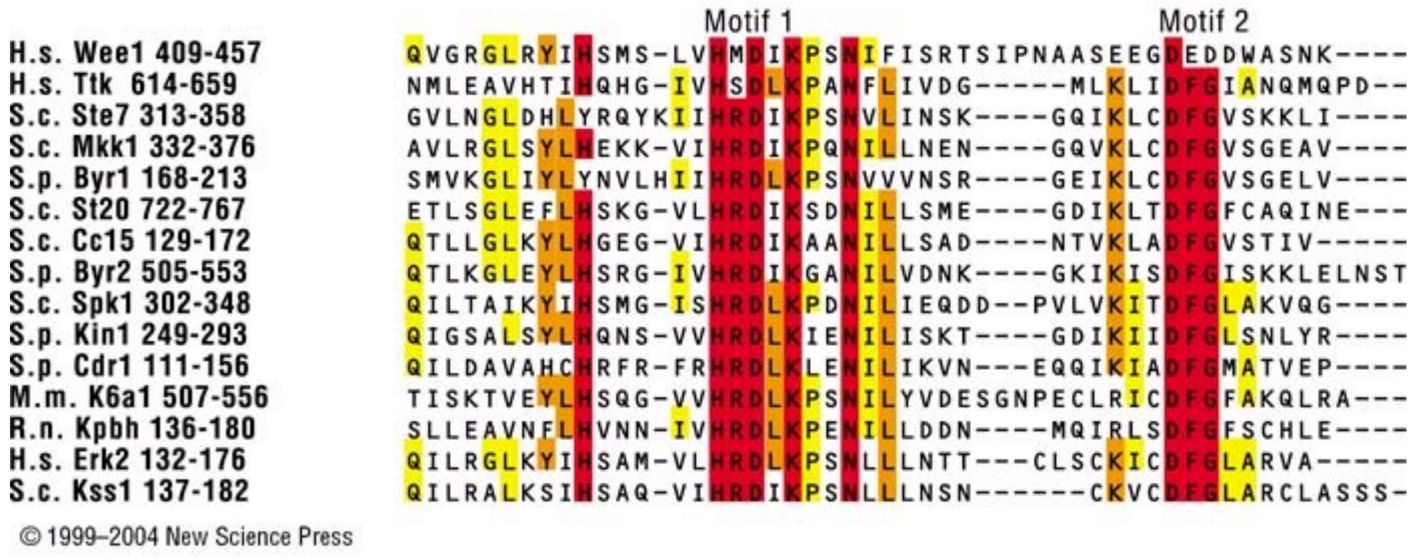
```
Reference  N I T N L C P F G E V F N A T R F A S V Y A W N R K R I S N C V A D Y S V L Y N S A S F S T F K
Alpha      N I T N L C P F G E V F N A T R F A S V Y A W N R K R I S N C V A D Y S V L Y N S A S F S T F K
Beta       N I T N L C P F G E V F N A T R F A S V Y A W N R K R I S N C V A D Y S V L Y N S A S F S T F K
Gamma      N I T N L C P F G E V F N A T R F A S V Y A W N R K R I S N C V A D Y S V L Y N S A S F S T F K
Delta      N I T N L C P F G E V F N A T R F A S V Y A W N R K R I S N C V A D Y S V L Y N S A S F S T F K
Omicron    N I T N L C P F D E V F N A T R F A S V Y A W N R K R I S N C V A D Y S V L Y N L A P F F T F K
                             10                  20                  30                  40

Reference  C Y G V S P T K L N D L C F T N V Y A D S F V I R G D E V R Q I A P G Q T G K I A D Y N Y K L P
Alpha      C Y G V S P T K L N D L C F T N V Y A D S F V I R G D E V R Q I A P G Q T G K I A D Y N Y K L P
Beta       C Y G V S P T K L N D L C F T N V Y A D S F V I R G D E V R Q I A P G Q T G N I A D Y N Y K L P
Gamma      C Y G V S P T K L N D L C F T N V Y A D S F V I R G D E V R Q I A P G Q T G T I A D Y N Y K L P
Delta      C Y G V S P T K L N D L C F T N V Y A D S F V I R G D E V R Q I A P G Q T G K I A D Y N Y K L P
Omicron    C Y G V S P T K L N D L C F T N V Y A D S F V I R G D E V R Q I A P G Q T G N I A D Y N Y K L P
             50                  60                  70                  80                  90

Reference  D D F T G C V I A W N S N N L D S K V G G N Y N Y L Y R L F R K S N L K P F E R D I S T E I Y Q
Alpha      D D F T G C V I A W N S N N L D S K V G G N Y N Y L Y R L F R K S N L K P F E R D I S T E I Y Q
Beta       D D F T G C V I A W N S N N L D S K V G G N Y N Y L Y R L F R K S N L K P F E R D I S T E I Y Q
Gamma      D D F T G C V I A W N S N N L D S K V G G N Y N Y L Y R L F R K S N L K P F E R D I S T E I Y Q
Delta      D D F T G C V I A W N S N N L D S K V G G N Y N Y R Y R L F R K S N L K P F E R D I S T E I Y Q
Omicron    D D F T G C V I A W N S N X L D S K V S G N Y N Y L Y R L F R K S N L K P F E R D I S T E I Y Q
                100                 110                 120                 130                 140

Reference  A G S T P C N G V E G F N C Y F P L Q S Y G F Q P T N G V G Y Q P Y R V V V L S F E L L H A P A T V
Alpha      A G S T P C N G V E G F N C Y F P L Q S Y G F Q P T Y G V G Y Q P Y R V V V L S F E L L H A P A T V
Beta       A G S T P C N G V K G F N C Y F P L Q S Y G F Q P T Y G V G Y Q P Y R V V V L S F E L L H A P A T V
Gamma      A G S T P C N G V K G F N C Y F P L Q S Y G F Q P T Y G V G Y Q P Y R V V V L S F E L L H A P A T V
Delta      A G S K P C N G V E G F N C Y F P L Q S Y G F Q P T N G V G Y Q P Y R V V V L S F E L L H A P A T V
Omicron    A G N K P C N G V A G F N C Y F P L R S Y S F R P T Y G V G H Q P Y R V V V L S F E L L H A P A T V
                 150                 160                 170                 180                 190
```

Contacting residues between SARS-CoV-2 and ACE2. Boxes denote the contacting residues. Black boxes denote mutations unique to omicron, red boxes denote mutations occurring in multiple variants, and grey boxes denote no mutations in any variant.

Image credit: Li et al.
doi: https://doi.org/10.1101/2021.12.08.471688

From pairwise to multiple

- A multiple sequence alignment of k sequences is an insertion of gaps in the positions of the sequences, just like a pairwise alignment.



```
                                         Motif 1                  Motif 2
H.s. Wee1 409-457    QVGRGLRYIHSMS-LVHMDIKPSNIFISRTSIPNAASEEGDEDDWASNK----
H.s. Ttk  614-659    NMLEAVHTIHQHG-IVHSDLKPANFLIVDG-----MLKLIDFGIANQMQPD--
S.c. Ste7 313-358    GVLNGLDHLYRQYKIIHRDIKPSNVLINSK----GQIKLCDFGVSKKLI----
S.c. Mkk1 332-376    AVLRGLSYLHEKK-VIHRDIKPQNILLNEN----GQVKLCDFGVSGEAV----
S.p. Byr1 168-213    SMVKGLIYLYNVLHIIHRDLKPSNVVVNSR----GEIKLCDFGVSGELV----
S.c. St20 722-767    ETLSGLEFLHSKG-VLHRDIKSDNILLSME----GDIKLTDFGFCAQINE---
S.c. Cc15 129-172    QTLLGLKYLHGEG-VIHRDIKAANILLSAD----NTVKLADFGVSTIV-----
S.p. Byr2 505-553    QTLKGLEYLHSRG-IVHRDIKGANILVDNK----GKIKISDFGISKKLELNST
S.c. Spk1 302-348    QILTAIKYIHSMG-ISHRDLKPDNILIEQDD--PVLVKITDFGLAKVQG----
S.p. Kin1 249-293    QIGSALSYLHQNS-VVHRDLKIENILISKT----GDIKIIDFGLSNLYR----
S.p. Cdr1 111-156    QILDAVAHCHRFR-FRHRDLKLENILIKVN---EQQIKIADFGMATVEP----
M.m. K6a1 507-556    TISKTVEYLHSQG-VVHRDLKPSNILYVDESGNPECLRICDFGFAKQLRA---
R.n. Kpbh 136-180    SLLEAVNFLHVNN-IVHRDLKPENILLDDN----MQIRLSDFGFSCHLE----
H.s. Erk2 132-176    QILRGLKYIHSAM-VLHRDLKPSNLLLNTT---CLSCKICDFGLARVA-----
S.c. Kss1 137-182    QILRALKSIHSAQ-VIHRDIKPSNLLLNSN------CKVCDFGLARCLASSS-
```

- “Two homologous sequences whisper, a multiple alignment shouts loudly” -- Arthur M. Lesk

Heuristic Algorithm for Multiple Alignment

- There is a simple algorithm that can merge pairwise alignments to a multiple alignment.
- The algorithm does not guarantee the optimality of the result. But runs relatively fast.

Merging Two Pairwise Alignments

- Suppose we have the following two pairwise alignments
- ` t:  A-GAGC`
- `s1:  ATGAGC`
- and
- ` t:  AGA-GC`
- `s2:  AGTTGC`
- Our main idea is to use the shared sequence t to construct a multiple alignment of s1, t, and s2.

Merging Two Pairwise Alignments

```
 t:   A-GAGC
s1:   ATGAGC
and
 t:   AGA-GC
s2:   AGTTGC
```

```
 t:   A-GA-GC
s1:   ATGA-GC
and
 t:   A-GA-GC
s2:   A-GTTGC
```

```
 t:   A-GA-GC
s1:   ATGA-GC
s2:   A-GTTGC
```

- Algorithm: Insert gaps to the two alignments, so that the superstrings for t become the same in the two alignments. Then put the two alignments together.
- Property: Maintains the pairwise alignment unchanged if ignoring the all-gap columns of the pairwise alignment.

Merging Two Pairwise Alignments

```
 t:   A-GA-GC
s1:   ATGA-GC
s2:   A-GTTGC

 t:   AGA-GC
s3:   ATA--C
```

```
 t:   A-GA-GC
s1:   ATGA-GC
s2:   A-GTTGC

 t:   A-GA-GC
s3:   A-TA--C
```

```
 t:   A-GA-GC
s1:   ATGA-GC
s2:   A-GTTGC
s3:   A-TA--C
```

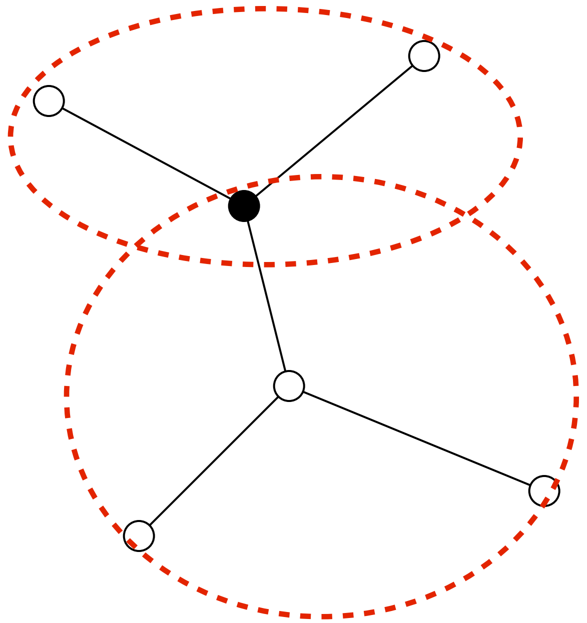
- The process can be continued to merge two multiple alignments together as a bigger one.
- Note that the obtained multiple alignment may not be optimal.

Heuristic Algorithm for Multiple Alignment

- Input: $s_1, s_2, \ldots, s_n$.
- Algorithm:
  - Let A = pairwise alignment of s1 and s2.
  - For i from 3 to n.
    - Construct pairwise alignment P between s1 and si.
    - Let A = merge(A, P, s1), i.e., merging A and P using s1 as the template.
  - Return A.

Progressive Alignment

- The order of the merging is important to get good (but not optimal) multiple alignment.
- We want to merge similar sequences first.
- One way is to construct a minimum spanning tree, and then merge using the shared vertices.

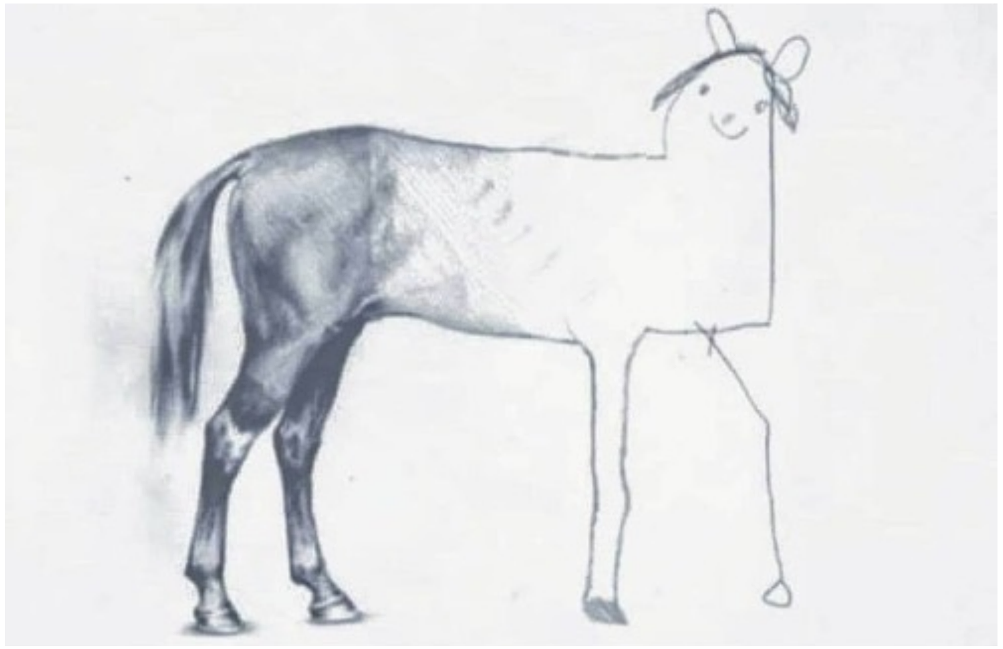


Heuristic Algorithm

- The exact algorithm for multiple alignment is super-polynomial. Before we study it, let’s examine a heuristic algorithm.
- A heuristic algorithm is an algorithm that gives up quality for speed. It usually does not offer any performance guarantee in terms of quality.
- Well… We already sacrificed the quality because of a simple scoring function any way.
- If we cannot afford exponential time, the best we can ask for is a suboptimal solution.
- But practically, it might work sometimes.

Heuristic Algorithm

- We do not want to spend super-polynomial (e.g. exponential) time.



“You get what you pay for”.

Exact Algorithm for Multiple Alignment

- When the optimal alignment is needed. There is an exact algorihtm as well.

Score Function

- For each column, assuming the letters are $a_1, \ldots, a_n$ for the n sequences. Define a column score $S(a_1, \ldots, a_n)$.
- The alignment score is the total of the column score.
- There are more than one ways to define the column score. Let us examine the simplest one first.

SP-score

- SP (Sum of Pair) score is the most widely used because its simplicity.
- We have a similarity matrix s(a,b) for any two given letters.
- For the k-th column with n letters $x_1, \ldots, x_n$. Define

$$S_k = \sum_{i,j} s(x_i, x_j)$$

- The SP-score of the alignment is

$$\sum_{1 \le k \le m} S_k$$

SP-score

- Note that SP-score can be viewed from a different perspective as follows.
- For each pair of sequences Si and Sj (including the gaps) in the multiple sequence alignment, define
- $S_{i,j} = \sum_k s\left(S_i[k], S_j[k]\right)$
- The SP score is then $\sum_{i,j} S_{i,j}$
- The two perspectives are equivalent if s(–,–)=0.

Review

- Before we study the algorithm to do multiple sequence alignment, we review the pairwise alignment algorithm.

```
Case 1:

S1[1..i]          -
S2[1..j-1]     s2[j]
```

```
Case 2:

S1[1..i-1]     s1[i]
S2[1..j]          -
```

```
Case 3:

S1[1..i-1]     s1[i]
S2[1..j]       s2[j]
```

$$DP[i,j] = \max \begin{cases} DP[i-1, j-1] + f(s[i], t[j]); \\ DP[i-1, j] + f(s[i], -); \\ DP[i, j-1] + f(-, t[j]); \end{cases}$$

Alignment of three sequences

- Use X to indicate a letter. The last column has 7 cases:

X	-	X	-	X	-	X
X	X	-	-	X	X	-
X	X	X	X	-	-	-

- Let f be the column-wise score function.
- DP[i_1, i_2, i_3] is the optimal score for $s_1[1..i_1]$, $s_2[1..i_2]$, $s_3[1..i_3]$.
- DP[i_1, i_2, i_3] = maximum of the 7 cases

case XXX: $DP[i_1, i_2, i_3] = DP[i_1-1, i_2-1, i_3-1] + S(s_1[i_1], s_2[i_2], s_3[i_3])$

case XX–: $DP[i_1, i_2, i_3] = DP[i_1-1, i_2-1, i_3] + S(s_1[i_1], s_2[i_2], -)$

……

- It is necessary to introduce some notation for n sequences …

Alignment of many sequences

X	-	X	-	X	-	X
X	X	-	-	X	X	-
X	X	X	X	-	-	-

- Here X indicates a letter and a dash indicates an indel.
- If you regard X as 1 and - as 0. Then these cases are all the 3-bit binary numbers but 000.
- In general, for n sequences, there are 2^n-1 cases.
- Let us use δ_j to indicate whether at sequence j the last column is a letter. $\delta_j = 1$ if it is a letter.

Algorithm for Optimal Multiple Alignment

- Let $S(x_1, \ldots, x_n)$ be a function to compute the column score.
- Let $D[i_1,\ldots,i_n]$ be the optimal multiple alignment score of $s_1[1..i_1],\ldots, s_n[1..i_n]$.
- Consider the last column of the optimal multiple alignment. There are 2^n-1 cases.
- For each case, the "sub-alignment" after removing the last column is also optimal.
- So $D[i_1,i_2,\ldots,i_n] = \max_{\delta_j=0,1}(D[i_1-\delta_1,\ldots,i_n-\delta_n]+S(b_1,\ldots,b_n))$
- where

$$b_j = \begin{cases} -, & \text{if } \delta_j = 0 \\ s_j[i_j], & \text{if } \delta_j = 1 \end{cases}$$

Complexity

- Time complexity $O(m^n 2^n T)$, where T is the time needed for computing a column score.
- Space complexity $O(m^n)$.
- This problem is NP-hard. So, no polynomial time algorithm exists (unless P=NP).
- Computing the optimal alignment with large m and n is hopeless. But at least this works for small m and n.

Better Scoring Function

- SP score (either maximization or minimization) is one of the simplest scoring function in use.
- There are better proposals to the scoring function.
- Let’s examine a more sophisticated score called relative entropy. We want to maximize.

Score 2: Relative Entropy

- There are n letters $x_1, \ldots, x_n$ in a column. For letter a in alphabet, let n_a be the number of a in the n letters.
- Our two different models assume two different distributions of letters in that column.
- Model 1: a occurs with probability $p_a = n_a / n$.
- Model 2: a occurs with "background" probability q_a, which is learned from a database.
- We want to compute the likelihood ratio.

Relative Entropy

- Log likelihood ratio:

$$\Pr(x_1,\ldots,x_n \mid \text{model1}) = \prod_{1\le i\le n} p_{x_i}$$

$$\Pr(x_1,\ldots,x_n \mid \text{model2}) = \prod_{1\le i\le n} q_{x_i}$$

$$\log \prod_{1\le i\le n} \frac{p_{x_i}}{q_{x_i}} = \sum_{1\le i\le n} \log \frac{p_{x_i}}{q_{x_i}} = \sum_{a\in\Sigma} n_a \log \frac{p_a}{q_a}$$

- This is called the relative entropy at a column.
- Let E_j be the relative entropy of column j.
- The alignment score is equal to $\sum_{1\le j\le m} E_j$
- We want to maximize.

Relative Entropy

- A few assumptions of relative entropy score:
  - Sequences are independent to each other.
  - Columns are independent to each other.
- These seem to be too strong but at least when these assumptions are true the score has a theory foundation. Other scores are more empirical.
- Note: If all q_a are equal to each other, then this is equivalent to the “minimum entropy” introduced in the reference book (Durbin page 138).

Approximation Algorithm

- In the heuristic algorithm, it would be good if we can get some guarantee on the quality of the suboptimal result.
- Suppose we want to maximize a score, and the optimal value is OPT, how about computing a solution with score at least OPT/c for a small constant $c>1$?
  - If $c = 1.001$ this is not so bad.
- If this can be done, this is called a ratio-c approximation algorithm for the problem.
- For minimization problem, ratio c means the algorithm gives score at most c*OPT.

The Minimization Version of Multiple Alignment

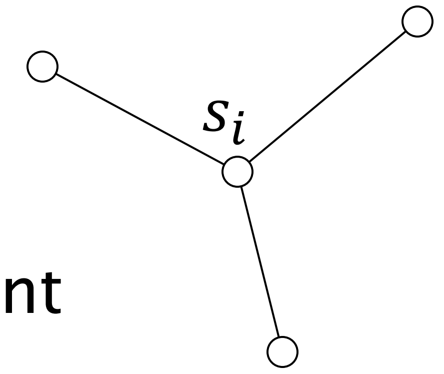
- The known approximation algorithm for multiple alignment only works for the minimization version of the problem.
- Let M[a,b] be a distance metric between two letters (possibly –) a and b satisfying **triangular inequality**.
  - M[a,a] = 0
  - M[a,b] >=0
  - M[a,c] <= M[a,b]+M[b,c]
- The distance between two sequences $d'(S_i, S_j)$ is the total of their column scores using M as the score scheme.
- The SP score is $\sum_{i,j} d'(S_i, S_j)$.
- We want to **minimize**.
- Note that this is almost the same as before, except that now we want to minimize the difference.

Triangular Inequality

- For sequences S,T,R taken from the same multiple alignment
  - d'(S,R) <= d'(S,T) + d'(T,R)
- Note that S, T, R may have the '–' symbols in the sequence.
- Notations:
  - When context is clear (the multiple alignment is given), we also use d'(s,t) to denote the distance between the original sequences s and t (without '–'). It is equal to d'(S,T).
  - When there are more than one multiple alignments $A_1, A_2, \ldots, A_n$, we use $d_i'(s, t)$ to indicate the distance induced by A_i.
  - We use d(s,t) to denote the optimal pairwise alignment between s and t. Note that d(s,t)<= d'(s,t)

Approximation Algorithm

- Consider the following algorithm:
- Input: $s_1, s_2, \ldots, s_n$
- 1. Select an s_i. Build optimal pairwise alignment between s_i and s_j for each $j=1,2,\ldots,n$.
- 2. Now use these pairwise alignment to build a multiple alignment A_i.
- 3. Repeat 1 and 2 for every i, and output the A_i with the minimum SP score.



We want to prove that this algorithm has ratio 2.

Approximation Ratio

- Fact 1: $d'_i(s_i, s_j) = d(s_i, s_j)$ for every j.

Approximation Ratio

- Let A* be the optimal multiplke alignment and d* be pairwise alignment score induced by optimal multiple alignment

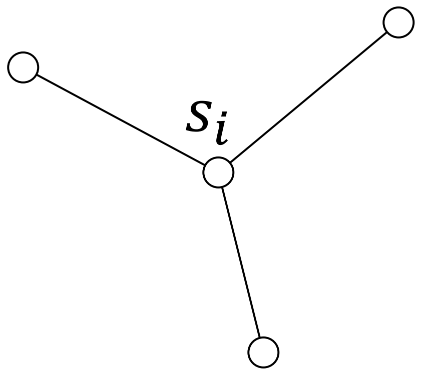
$$\sum_{1\le i\le n} score(A_i) = \sum_{1\le i\le n}\sum_{1\le k\le n}\sum_{1\le l\le n} d'_i(s_k,s_l) \le \sum_{1\le i\le n}\sum_{1\le k\le n}\sum_{1\le l\le n}(d'_i(s_i,s_k)+d'_i(s_i,s_l))$$

$$\le 2n\sum_{1\le i\le n}\sum_{1\le k\le n} d'_i(s_i,s_k)$$

$$= 2n\sum_{1\le i\le n}\sum_{1\le k\le n} d(s_i,s_k)$$

$$\le 2n\sum_{1\le i\le n}\sum_{1\le k\le n} d^*(s_i,s_k)$$

$$= 2n \times OPT$$



- Recall that
  - d'_i: pairwise alignment score induced by A$_i$
  - d: optimal pairwise alignment score
- So, at least one of the A_i has score no more than 2OPT.

Approximation Algorithm

- Theorem: The above algorithm is a ratio-2 approximation to multiple alignment under SP score.
- Note: For general score scheme (general score matrix) this ratio-2 may not hold.
    - For example, when the score scheme does not satisfy triangular inequality. Or when the score scheme has both positive and negative scores.

Applications

- Motif finding: conserved regions of a protein family may be important motifs.
- Structure prediction: after multiple alignment of a protein family, their structures can be predicted together.
- Phylogeny: the pairwise alignment induced by the multiple alignment can be more accurate than the optimal pairwise alignment in reality. Build multiple alignment first then do phylogeny by examining the evolution on each column.

What we have learned?

- 1. Multiple sequence alignments “shout out”.
- 2. extend the DP algorithm to multiple alignment
- 3. first NP-hard problem in this course
- 4. two ways to handle NP-hard problems: heuristics & approximation
- 5. apply our likelihood ratio score to multiple alignment